# Sequence Diagrams

Massimo Felici

JCMB-1402 0131 650 5899

1BP-G04 0131 650 4408

mfelici@inf.ed.ac.uk

# What are Sequence Diagrams?

- Interactions Diagrams
  - **Sequence diagrams**
  - **Interaction overview diagrams**
  - **Timing diagrams**
  - **Communication diagrams**
- Interaction diagrams model important runtime interactions between the parts that make up the system
- **Sequence Diagrams** are interaction diagrams that detail how **operations** are carried out

# What Do Sequence Diagrams Model?

- capture the interaction between **objects** in the context of a **collaboration**
- show object instances that play the roles defined in a collaboration
- don't show the structural relationships between objects
- show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when

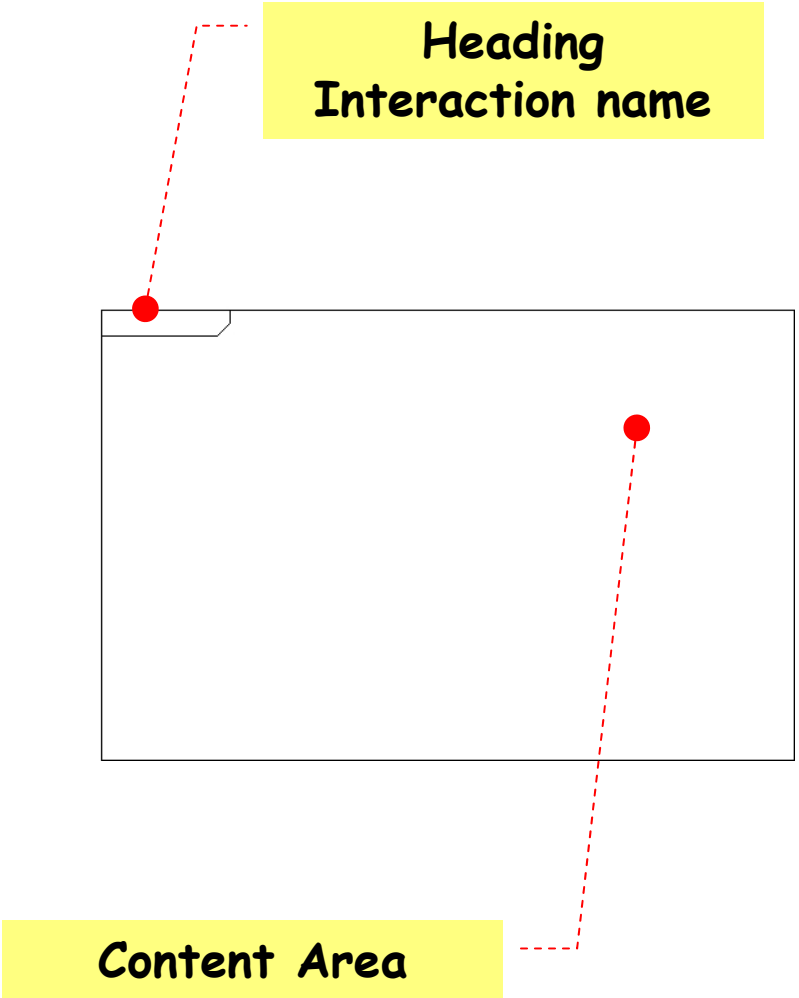
# Participants in a Sequence Diagram

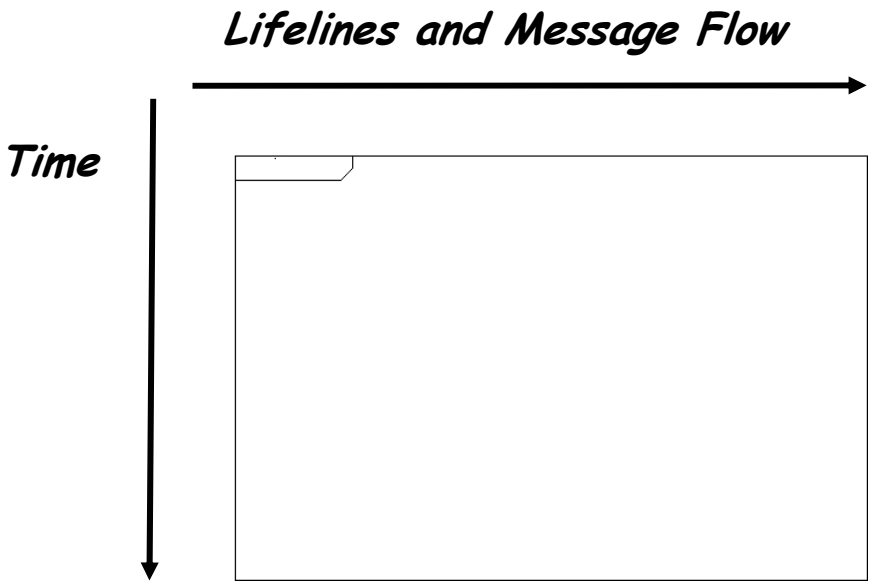
- A **sequence diagram** is made up of a collection of **participants**
- Participants - the system parts that interact each other during the sequence
- **Classes** or **Objects**: each object (class) in the interaction is represented by its named icon along the top of the diagram

# Sequence Diagrams at a Glance

- Sequence Diagrams show elements as they interact over time, showing interactions or interaction instances
- Notations
  - Frames, Lifelines, Messages and Focus Control, Combined Fragments, Interaction Occurrences, States, Continuations, Textual Annotation and Tabular Notation

# Frames

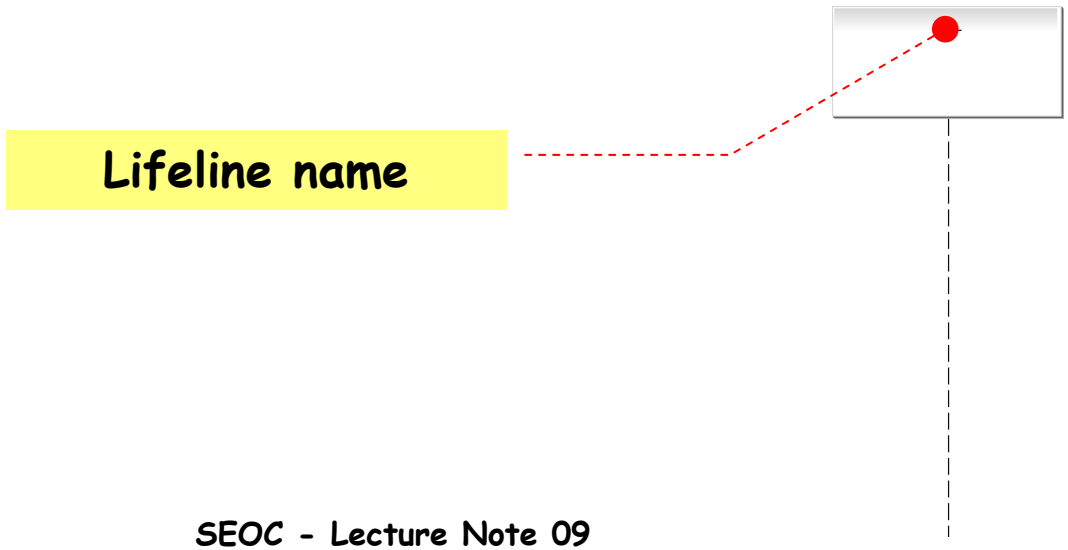

Heading
Interaction name
Content Area


Lifelines and Message Flow
Time

# Lifelines

- Sequence diagrams are organized according to time
- Each participant has a corresponding lifeline
- **Lifelines**: each vertical dotted line is a lifeline, representing the time that an object exists
- **Lifeline name**

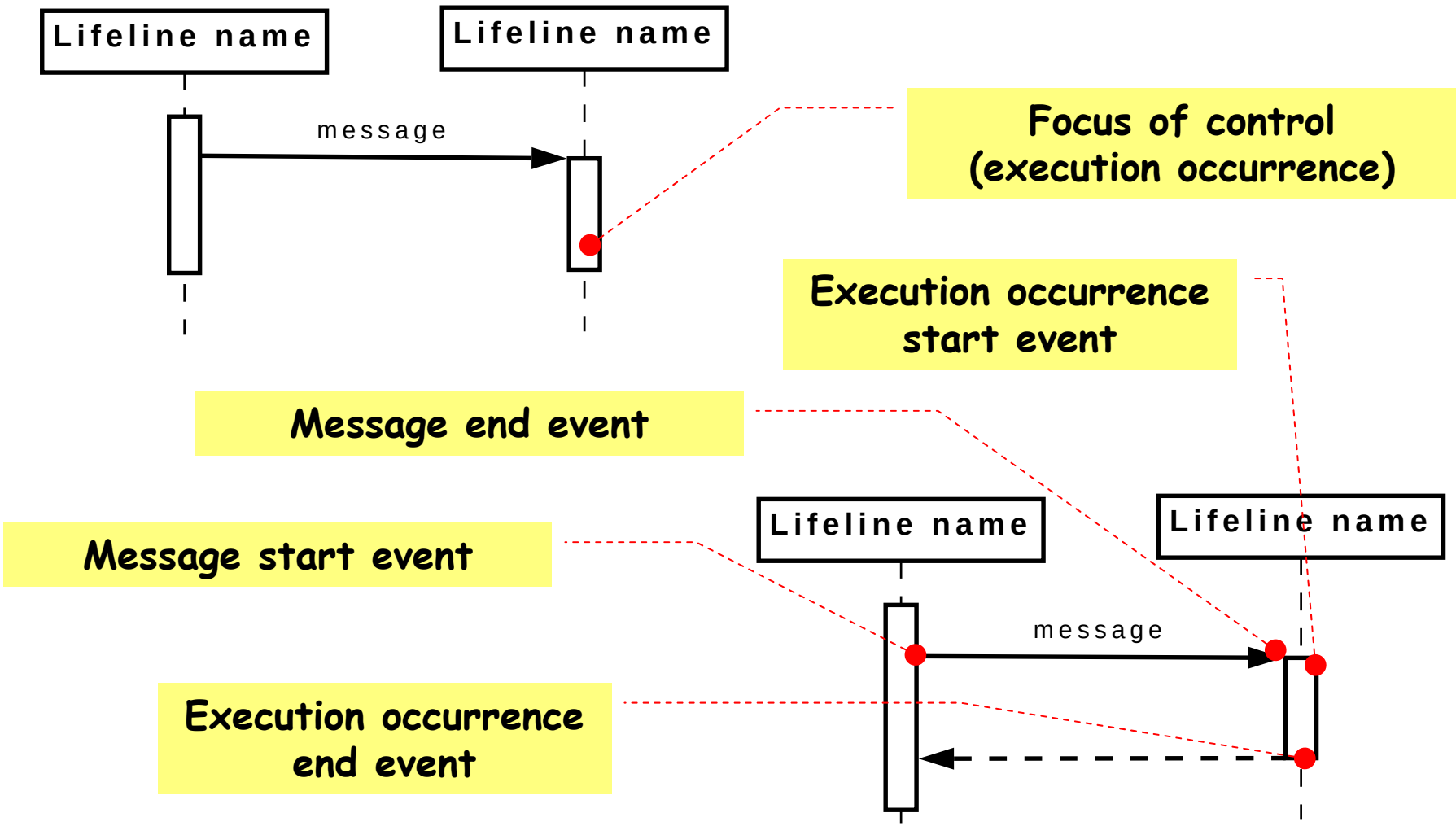
  **[connectable-element-name][`[`selector']'][:class-name][decomposition]**



# Examples of lifeline names

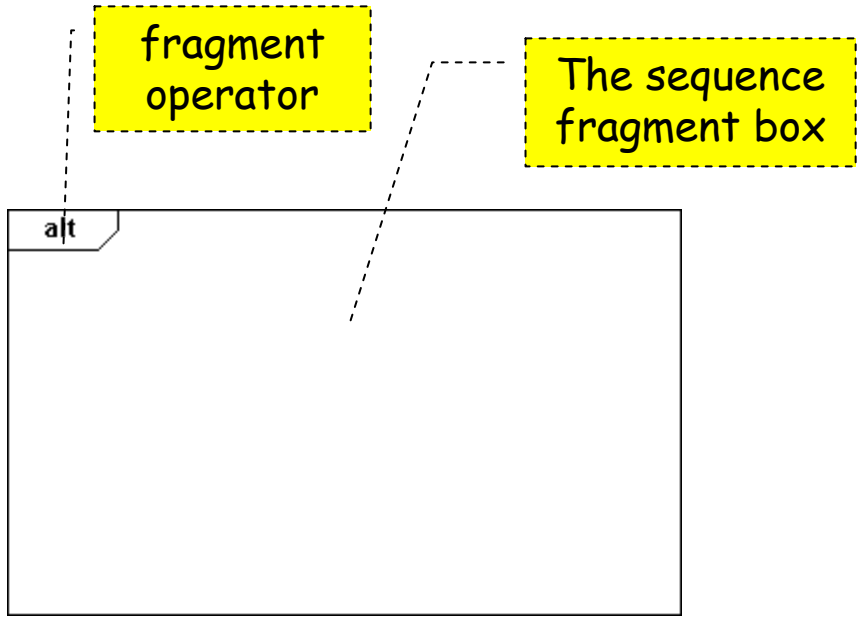
| Syntax | Explanation |
|---|---|
| seoclecturer | An object named secolecturer |
| seoclecturer : Lecturer | An object names seoclecturer of class Lectuer. |
| :Lecturer | An anonymous object of class Lecturer |
| lecturer[i] | The object lecturer that is selected by the index value *i*. |
| s ref sd3 | A subsystem s whose internal interaction is shown in sequence diagram sd3 (decomposition). |
| self | The connectable element that owns the interaction shown in the sequence diagram |

# Messages and Focus of Control



# Sequence Fragments

- UML 2.0 introduces Sequence (or Interaction) Frames
- A sequence fragment is represented as a box, called a **combined fragment**, which encloses a portion of the interactions within a sequence diagram
- The **fragment operator** (in the top left cornet) indicates the type of fragment
- Fragment types: **ref**, **assert**, **loop**, **break**, **alt**, **opt**, **neg**




- Sequence fragments make it easier to create and maintain accurate sequence diagrams

# Messages

- **Messages** (or signals) on a sequence diagram are specified using an arrow from the participant (message caller) that wants to pass the message to the participant (message receiver) that is to receive the message
- A **Message** (or stimulus) is represented as an arrow going from the sender to the top of the focus of control (i.e., execution occurrence) of the message on the receiver's lifeline

# Message Type Notation

| | | | |
|---|---|---|---|
| Synchronous<br>Or Call | Asynchronous | Creation | Reply (Return) |

# Message and Argument Syntax

- Message Syntax

[attribute=] signal-or-operation-name [(argument)] [:return-value]|*

- Argument syntax

[parameter-name=] argument-value| attribute=out-parameter-name [:argument-value] | -

# Creation and Destruction Messages

- **Element Creation**: when an element is created during an interaction, the communication that creates the element is shown with its arrowhead to the element

- **Element Destruction**: When an element is destroyed during an interaction, the communication that destroys the element is shown with its arrowhead to the element's lifeline where the destruction is marked with a large **X** symbol

# Combined Frames

- It is possible to combine frames in order to capture, e.g., loops or branches.
- **Combined fragment keywords**: alt, opt, brak, par, seq, strict, neg, critical, ignore, consider, assert and loop
- Other ways in UML 2.0 of hiding information are by **interaction occurrences** and **continuations**

# Other notations

- **States** – it is possible to place states on lifelines (e.g., pre and post conditions)
- **Textual notations** (e.g., comments, time constraints, duration constraints)
- **Tabular notation**

# Timing

- **Constraints** are usually used to show timing constraints on messages. They can apply to the timing of one message or intervals between messages.
- **Durations**. The duration of activations or the time between messages can be show with construction marks.

# How to Produce Sequence Diagrams

1. **Decide on Context**: Identify **behavior** (or **use case**) to be specified
2. **Identify structural elements**:
   1. Model **objects** (classes)
   2. Model **lifelines**
   3. Model **activations**
   4. Model **messages**
   5. Model **Timing constraints**
3. Refine and elaborate as required

# How do interaction diagrams help?

- **Check use cases**
- **Check class can provide an operation**
  - showing how a class realizes some operation by interacting with other objects
- **Describe design pattern**
  - parameterizing by class provides a scheme for a generic interaction (part of Software Architecture)
- **Describe how to use a component**
  - capturing how components can interact

# Readings

- **UML course textbook**
  - Chapter 9 on Interaction Sequence Diagrams

# Summary

- **Sequence Diagrams**
  - capture some elements of the dynamics of systems
  - Support a number of different activities
  - Describe interaction in some detail, including timing
- **Dimensions**
  - Objects and Time
- **Basics**
  - Objects, Lifelines, Activations, Messages, etc.
- **Timing**